Where is the Marble?

Link:
https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1415

Solution:

C++	https://ideone.com/Vmp23i
Java	https://ideone.com/IJl0Uo
Python	https://ideone.com/FkB8HE

Tóm tắt đề: Cho N viên bi, trên mỗi viên bi có các con số. Raju và Meena sắp xếp lại các viên bi theo thứ tự tăng dần các con số ghi trên viên bi. Raju cần phải tìm vị trí của viên bi đầu tiên mà con số ghi trên đó đúng bằng một giá trị cho trước. Vị trí được tính từ 1 đến N.

Input:

Gồm nhiều test case, mỗi test case có cấu trúc như sau:

- Dòng đầu chứa 2 số nguyên dương N và Q là số viên bi và số câu hỏi cần trả lời.
- N dòng tiếp theo, mỗi dòng chứa một số nguyên là các số ghi trên từng viên bi.
- Q dòng cuối cùng, mỗi dòng chứa một số x là con số cần tìm ở câu hỏi thứ i.

Có ít hơn 65 test case và các số trong mỗi test case đều không âm và không lớn hơn 10000. Input kết thúc bởi cặp N = 0 và Q = 0.

Output:

Với mỗi test case, in ra theo cấu trúc sau:

- Dòng đầu là “CASE# X” với X là số thứ tự của test case (tính từ 1).
- Q dòng tiếp theo, mỗi dòng sẽ ứng với câu trả lời của từng câu hỏi trong Q câu hỏi. Nếu tìm thấy số X xuất hiện đầu tiên ở viên bi vị trí Y, thì xuất ra “X found at Y”; ngược lại nếu không tìm thấy giá trị X trong mảng thì xuất ra “X not found”.

4 1 2 3 5	CASE# 1: 5 found at 4 CASE# 2: 2 not found

Big-O Coding

1 5 5 2 1 3 3 3 1 2 3 0 0	3 found at 3

Giải thích:

- Ở test 1, mảng sau khi sắp xếp tăng dần là {1, 2, 3, 5}. X=5 xuất hiện lần đầu tại vị trí thứ 4, nên xuất ra "5 found at 4".
- Ở test 2, mảng sau khi sắp xếp tăng dần là {1, 1, 3, 3, 3}. Ở câu hỏi thứ nhất, X = 2 không xuất hiện trong mảng, nên xuất ra "2 not found". Ở câu hỏi thứ hai, X = 3 xuất hiện ở 3 vị trí là 3, 4 và 5, nhưng vị trí xuất hiện đầu tiên là 3 nên kết quả là "3 found at 3"

Hướng dẫn giải: Ta sắp xếp lại mảng tăng dần. Sau đó sử dụng hàm lower_bound hoặc tự cài đặt hàm BS_first để tìm vị trí đầu tiên xuất hiện trong mảng là được. Cần lưu ý ở chỗ index tính từ 0 hoặc 1 đến tránh sai lệch.

Độ phức tạp: O(T * Q * logN) với T là số lượng test, Q là số lượng truy vấn và N là số lượng viên bi.

Big-O Coding